IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Applicant(s):	Jai et al.
Docket No:	5-4-52
Serial No.:	10/600,995
Filing Date:	June 20, 2003
Group:	2446
Examiner:	Benjamin R. Bruckart
Title:	Automated Transformation of Specifications for Devices into Executable Modules

---

REPLY BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Appellants hereby reply to the Examiner's Answer, mailed October 30, 2009 (referred to hereinafter as "the Examiner's Answer"), in an Appeal of the final rejection of claims 1 through 21 in the above-identified patent application.

REAL PARTY IN INTEREST

A statement identifying the real party in interest is contained in Appellants' Appeal Brief.

RELATED APPEALS AND INTERFERENCES

A statement identifying related appeals is contained in Appellants' Appeal Brief.

STATUS OF CLAIMS

A statement identifying the status of the claims is contained in Appellants' Appeal Brief.

STATUS OF AMENDMENTS

A statement identifying the status of the amendments contained in Appellants' Appeal Brief is not correct. The amendments filed on December 29, 2008 have been entered.

SUMMARY OF CLAIMED SUBJECT MATTER

A Summary of the Invention is contained in Appellants' Appeal Brief.

STATEMENT OF GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A statement identifying the grounds of rejection to be reviewed on appeal is contained in Appellants' Appeal Brief.

CLAIMS APPEALED

A copy of the appealed claims is contained in an Appendix of Appellants' Appeal Brief.

ARGUMENT

The Examiner asserts that the "definition of executable statements" is *the* subject of the appeal.

Appellants note that the "definition of executable statements" is just one of the subjects of the appeal argued in the Appeal Brief and Reply Brief. (See arguments below.)

Point 1

The Examiner asserts that the rule program is derived by compiling the rule file and operations file (paragraph [0058]). (Page 12 of the Examiner's Answer.)

Appellants note that Moir teaches, however, that the "*virtual machine compiler 60 <u>utilizes</u> the operations file 62 and the rule file 64 to <u>compile a rule program 66</u>.*" (Emphasis added.) The Examiner has changed this statement to assert that Moir teaches "*<u>compiling the operations file and rule files</u>.*" The statement "compiling a rule file" has a different meaning than "compiling a rule program" and "utilizes a rule file." The Examiner's statement implies that the rule file contains executable statements; the teaching in Moir does *not* imply that the rule file contains executable statements. Moir does *not* teach that the rule file is compiled and does *not* disclose or suggest that *one or more input rules comprise one or more* ***executable***

statements.

Point 2

The Examiner asserts that the configuration files are executed because, "otherwise, the configurations would not be referenced as dependent upon each other for each network device 'to perform a task' and that the configurations define 'behavior constraints' and 'behavior requirements' of each device." (Page 13 of the Examiner's Answer.)

Appellants note that the definition of "behavior constraints" and "behavior requirements" of each device does *not* require that the configuration files are executable; the configuration files may simply contain parameters that define the "behavior constraints" and "behavior requirements," as would be apparent to a person of ordinary skill in the art. Similarly, a reference that the configuration files are dependent upon each other for each network device "to perform a task" does *not* infer that the configuration files are executable.

Point 3

The Examiner asserts that the features upon which Applicant relies are not recited in the rejected claims. (Page 13 of the Examiner's Answer.)

Appellants note that at least one of the features relied upon, i.e., wherein said one or more input rules comprise one or more executable statements, is recited in the rejected claims.

Point 4

The Examiner asserts that Moir teaches the operations file and rules file contain statements, constraints, dependencies all of which are taken into consideration and executed by the virtual machine compiler when creating the rule program. (Page 13 of the Examiner's Answer.)

Appellants maintain that there is *no* teaching in Moir that the operations file and rules file are executed by the virtual machine compiler; Moir only teaches that the "*virtual machine compiler 60 <u>utilizes</u> the operations file 62 and the rule file 64 to <u>compile a rule program 66</u>.*"

Point 5

The Examiner equates processing operations files and rules files with a form of execution. (Page 13 of the Examiner's Answer.)

Appellants note that Moir teaches to *utilize* the operations files and rules files; Moir does *not* disclose or suggest *processing* operations files and rules files. In any case, processing files does *not* imply that the files are executed.

Point 6

The Examiner asserts that paragraph 72 (of Moir) is the most conclusive evidence that the rule has executable statements. (Page 14 of the Examiner's Answer.)

Appellants note that paragraph 72 of Moir specifically teaches that *the rule file 64 (including the if-then-else rules) is text that is* ***converted*** *into a binary-form rule program 66.* The cited paragraph therefore provides evidence that the rule file contains *text* and *not executable statements,* and that the rule file *must be converted* into a program in order to be executable.

Point 7

The Examiner asserts that Appellant "brings in another definition of executable statement as 'performs an action'" and "the example gives examples of if/then statements and evaluation of statements." The Examiner asserts that such definition of executable statements reads directly on par 72 and 73 of Moir. (Page 14 of the Examiner's Answer.)

Appellants note that the Examiner has cited *only a portion* of the definition given on the bottom of page 7 of the Appeal Brief. For example, the cited definition clearly recites that an "executable statement" is a *statement that is executable*. As Moir teaches, the text of the rule files must be converted into a *binary-form rule program* in order to be executable. Thus, the text of the rule files is *not* executable.

Point 8

Regarding claim 4, the Examiner asserts, for example, that Moir illustrates a call chain by showing a rule invoking different actions and operations of other methods (paragraphs 72-73) based on the conditional statement and that a chain of calls is referenced by the rule file to perform said operations. (Page 15 of the Examiner's Answer.) Regarding claim 8, the Examiner asserts that a set of instance chain access for the given configuration are illustrated on the rule file from page 7 and the call chains are the methods and clauses (bottom of page 7) and operations (top of page 7).

Appellants note that a "call chain" is a chain of calls, as would be apparent to a person of ordinary skill in the art. A rule invoking different actions and operations of other methods based on the conditional statement and the methods and clauses (bottom of page 7) and operations (top of page 7) does *not* include a *chain of calls*, and thus does *not* include a "call chain." Also, contrary to the Examiner's assertion, the fact that "the component may have *multiple operations* with the same identifier but with different argument types" does *not* teach a "call chain," as defined in the context of the present invention, and does *not* teach *a set of instance chain accesses for the given configuration element*. Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule *a call chain emanating from the given rule,* and does *not* diselose or suggest *a set of instance chain accesses for the given configuration element*.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule a call chain emanating from the given rule, as required by claim 4, and do not disclose or suggest wherein the step of determining which of the plurality of configuration elements could be accessed further comprise the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element, as required by claim 8.

Point 9

Regarding claim 9, the Examiner points to page 6, paragraph 72, of Moir to show a "given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class" and asserts that a "class of an element is not defined in the claim." Therefore, the Examiner asserts that the class of element is taken to be the name of the network element which would be the name described on page 7. The Examiner further asserts that "every access for the other configuration class to other configuration element will be defined by the conditional statement and operations that are designated by name to the VOP file." (Page 15 of the Examiner's Answer.) In the previous Office Action, the Examiner asserted that Moir discloses wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements (Moir: page 8, continued from rule file page 7; invocation of more than operation with different arguments).

Appellants note that the present disclosure teaches:

> In an aspect of the invention, there are a number of configuration elements associated with a number of devices. Information about input configuration elements is accessed. An input configuration element is associated with one or more input rules. *A set of related configuration elements* *<u>associated with a device</u>* *may be also grouped into a configuration class.* For example, all information pertaining to an interface entry can be represented as a configuration class containing a number of configuration elements. Thus, the information is generally provided as a configuration class, but this is not necessary. In general, an input configuration element is a variable associated with a device and is part of specifications for that device. A configuration element may also be a configuration class or other complex element associated with a device.
> (Page 2, lines 13-23; emphasis added.)

Clearly, a patentee is entitled to be his own lexicographer. See, e.g., *Rohm & Haas Co. v. Dawson Chemical Co., Inc.*, 557 F. Supp 739, 217 U.S.P.Q. 515, 573 (Tex. 1983); *Loctite Corp. v. Ultraseal Ltd.*, 781 F.2d 861, 228 U.S.P.Q. 90 (Fed. Cir. 1985); and *Fonar Corp. v. Johnson & Johnson*, 821 F.2d 627, 3 U.S.P.Q.2d 1109 (Fed. Cir. 1987).

The interpretation of the term "configuration class" asserted by the Examiner is inconsistent with the definition provided in the specification and is not how the term would be understood by a person of ordinary skill, based on the specification. When the specification explains and defines a term used in the claims, without ambiguity or incompleteness, there is no need to search further for the meaning of the term. *Multiform Desiccants Inc. v. Medzam Ltd.*, 133 F.3d 1473, 45 U.S.P.Q.2d 1429, 1433 (Fed. Cir. 1998).

Also, contrary to the Examiner's assertion, "the invocation of more than operation with different arguments" does *not* teach *determining* *<u>every access</u>* *for the other configuration class to other configuration element.* Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest *determining* *<u>every access</u>* *for the other configuration class to other configuration element.*

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class, and wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements, as required by claim 9.

Point 10

The Examiner asserts that "the executable module is the virtual machine compiler that triggers (creates) the output rules corresponding to the input elements through deferred and direct triggering of the output rules." The Examiner further asserts that "the condition of deferred triggering is taught by Moir because the 'deferred rules are invoked as a sequence, typically at the end of a configuration session'" (page 8) and because the operations are not triggered until the rules program is loaded into the network connection device (page 5, paragraph 65). (Page 17 of the Examiner's Answer.) Regarding claim 14, the Examiner previously asserted that Moir discloses wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules (Moir: page 8, <operation> is similar to the definition in the spec that "deferred rules are invoked as a sequence, typically at the end of a configuration session").

First, Appellants note that a "trigger" is defined as to "activate." (See, dictionary.com) Contrary to the Examiner's assertion, Appellants note that the *creation* of output rules is *not* equivalent to the *triggering* of output rules, as would be apparent to a person of ordinary skill in the art. Second, contrary to the Examiner's assertion, Appellants find *no* mention of deferred rules invoked as a sequence on page 8 of Moir. Third, Appellants note that paragraph 65 of Moir is directed to compiling and loading a program; the paragraph does *not* mention and is *not* directed to *triggering output rules* or *triggering output rules corresponding to the given input configuration element through <u>deferred triggering</u>.* Finally, contrary to the Examiner's assertion, the <operations> disclosed by Moir do *not* teach that "deferred rules are invoked as a sequence, typically at the end of a configuration session" and do *not* teach *triggering output rules corresponding to the given input configuration element through <u>deferred triggering</u> of the one or more output rules*. Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules, as required by claim 14.

Point 11

The Examiner asserts that, "when a plurality of these (a batch) are created in the rules program, a batch of rules are triggered and directly created." (Page 17 of the Examiner's Answer.) Also, regarding claim 16, the Examiner previously asserted that Moir discloses wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules (Moir: page 8, continued from rule file page 7; batch of operations with the same identifier but with different argument types).

As noted above, contrary to the Examiner's assertion, the *creation* of output rules is *not* equivalent to the *triggering* of output rules, as would be apparent to a person of ordinary skill in the art. In addition, the operations with the same identifier (but with different argument types) disclosed by Muir do *not* teach *triggering output rules corresponding to the given input configuration element through* <u>*batch triggering*</u> *of the one or more output rules.* Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules, and does *not* disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules, as required by claim 14, and do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules, as required by claim 16.

Point 12

The Examiner asserts that Moir teaches direct triggering on page 9, paragraph 105 where the rule program may be directly loaded into the network device. (Page 17 of the Examiner's Answer.)

As the Examiner acknowledges, paragraph 105 of Moir is directed to loading the rule program; the cited paragraph does *not* mention and is *not* directed to the *direct triggering* of output rules.

Point 13

The Examiner asserts that Appellants have given no details about what direct pertains to and views the direct triggering as the output files being directly sent to the network device for which they are created to configure. The Examiner asserts that "the output rules are triggered when they are implemented because they dictate how the device operates" and "the output rules are directly triggered by the virtual machine compiler at the time of creation." (Page 17 of the Examiner's Answer.)

Appellants note that the term "direct" clearly refers to the triggering in the phrase "direct triggering." Appellants also note that the present specification teaches:

> (2) Direct rule. A direct rule is one that is invoked directly by configuration element modification operations. This triggering is effected automatically by executable modules.
> (Page 13, lines 11-13.)

See, also, FIG. 11. The present disclosure teaches that "the update methods performs direct triggering: *the rule is directly invoked as a method call from its body.*" (Page 16, lines 16-18; emphasis added.) The present disclosure also teaches that "*all invocation in the update method is directly invoked without delay. This is called direct triggering.*" (Page 23, lines 3-5; emphasis added.)

Thus, contrary to the Examiner's assertion, Appellants have given details about what direct pertains to. The Examiner's interpretation of "direct triggering" is inconsistent with the definition provided in the specification and is *not* how the term would be understood by a person of ordinary skill, based on the specification.

Point 14

The Examiner asserts that Presley teaches to look for cyclical relationships and other errors that can be construed as out-of-bounds or an invalid configuration. The Examiner further asserts that Presley teaches that by analyzing the configurations, you can find and correct configuration errors in order to provide reliable and predictable performance (Presley: page 1, paragraph 5). (Page 19 of the Examiner's Answer.) Also, regarding claim 17, the Examiner previously acknowledged that Moir fails to teach performing a circularity check, but asserts that

Presley teaches a method (that) further comprises the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself (page 4, paragraph 47) in order to provide reliable and predictable performance (page 1, paragraph 5).

In the text cited by the Examiner. Presley teaches:

> [0047] The core services 51 obtain the site-specific resource and component configuration XML doclets 53-55 by invoking probing service layers (not shown). The core services 51 analyze, advise, and correct out-of-bound and invalid configuration parameters and identify cyclic relationships. A set of new XML doclets 58 are generated and fielded to the individual network components 33 for implementation. The core services 51 also exports management, advising and alert services 56. The core services 51 can be exported through a browser service 57, as implemented on the management console 31 (shown in FIG. 2).

Thus, Presley teaches identifying cyclic relationships; Presley, however, does *not* teach performing a circularity check by *determining dependency relationships* *between two or more output rules* and by *determining* *whether a given one of the two or more output rules* *depends upon itself*.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself in order to provide reliable, as required by claim 17.

Appeal Brief Arguments

Independent Claims 1, 20 and 21

Independent claims 1, 20, and 21 were rejected under 35 U.S.C. §102(e) as being anticipated by Moir. Regarding claim 1, the Examiner asserts that Moir discloses generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements (page 5, paragraph 58)); and generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element (page 5, paragraph 60). In the Advisory Action, the Examiner asserts that the "configurations are executed to determine and control on how a device is to behave" (paragraph [0056]). The Examiner also asserts that "executable statements is broad and is not limited to code or a certain type of statement" and that "all the Moir reference has to show is that the statements are utilized or executed to perform an operation." The Examiner asserts that the rule program is derived by compiling the rule file and operations file (paragraph [0058]).

Appellants note that the present specification teaches that

> *input rules* are also part of specifications for a device and comprise, for example, *a set of checks or constraints or both* that should be performed before or after a configuration element is accessed. The input rules are generally derived from 'domain experts' (typically network specialists). An input rule is *usually represented as a set of executable statements*.
> (Page 2, lines 24-28.)

The present specification also teaches that

> *output rules* are *determined by using the accessed configuration elements, the input rules, and the way the input rule manipulates its accessed configuration elements*. Regarding the latter, output rules may be determined to deal with modifications to configuration elements, as explained in more detail below. In an illustrative embodiment, *each output rule is generally derived from exactly one input rule and corresponds to the same input configuration element associated with that input rule*. Output rules may be derived from multiple input rules, if desired.
> (Page 3, lines 7-14.)

Finally, the present disclosure teaches that

> an *executable module* is generated that is *adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element*.
> (Page 3, lines 15-17.)

In the text cited by the Examiner, however, Moir teaches:

> [0058] *The virtual machine compiler 60 utilizes the operations file 62 and the rule file 64 to compile a rule program 66*, which in one embodiment comprises a binary object including a sequence of instructions suitable for the virtual machine 10, discussed above. The rule program 66 comprises a set of operations, selected from operations supported by components of the network connection device 12, for performance by the respective components of the network connection device in accordance with the behavioral requirements defined by the rule file 64. In one embodiment, the rule program 66 may embody a number of sequences, these sequences constituting the classification rules 18, the event management rules 17 and the label management rules 19 discussed above with reference to FIG. 3.
> (Emphasis added.)

In the present Office Action, the Examiner asserts that paragraph 58 teaches that "the rule program is derived by compiling the rule file and operations file" (Office Action: page 3) and asserts that "the compiler binds processes behavior definitions and operations to data through compiling the operations file and rule files" (Office Action: page 10). Moir teaches, however, that the "*virtual machine compiler 60 utilizes the operations file 62 and the rule file 64 to compile a rule program 66*." (Emphasis added.) The Examiner has changed this statement to assert that Moir teaches "*compiling the operations file and rule files*." The statement "compiling a rule file" has a different meaning than "compiling a rule program." The Examiner's statement implies that the rule file contains executable statements; the teaching in Moir does *not* imply that the rule file contains executable statements. Moir does *not* teach that the rule file is compiled and does *not* disclose or suggest that *one or more input rules comprise one or more **executable statements***. Independent claims 1, 20, and 21 require *wherein said one or more input rules comprise one or more **executable statements***.

Regarding the Examiner's assertion that the "configurations are executed to determine and control on how a device is to behave" (paragraph [0056]), Appellants note that MOIR teaches:

> [0056] According to one embodiment of the present invention, a proposed solution to address the above identified network management problems includes *compiling the outcome of a number of discrete configuration steps* into an indivisible rule, which instructs a network device how to behave. This result may provide the advantage of allowing *configuration tasks* to be performed more reliably (and with a smaller code footprint), and also provides a mechanism for increasing the resolution of configuration without an adverse effect on the device's MTEF. Increased management resolution allows a network designer, for example, to safely exert control over very detailed aspects of the behavior of a network device, such as flow classification and data path features.
> (Emphasis added.)

Moir discloses *configuration steps*; Moir does not disclose or suggest, however, that *configuration files are executed.*

Regarding the Examiner's assertion that "executable statements is broad and is not limited to code or a certain type of statement" and that "all the Moir reference has to show is that the statements are utilized or executed to perform an operation," Appellants note that an "executable statement" is a statement that is executable, as would be apparent to a person of ordinary skill in the art. For example, MSDN (http://msdn.microsoft.com/enus/library/59b7dyw0(VS.80).aspx) teaches that:

> *An executable statement performs an action.* It can call a procedure, branch to another place in the code, loop through several statements, or evaluate an expression. An assignment statement is a special case of an executable statement.
> (Emphasis added.)

Moir, alternatively, teaches:

> [0057] FIG. 9 is a block diagram providing a high level diagrammatic representation of the operation of a virtual machine compiler 60, according to an exemplary embodiment of the present invention. The virtual machine compiler 60 is shown to receive as inputs: (1) *an operations file 62 that describes operations supported by components of a particular network device (i.e., component behavior) and constraint definitions, and (2) a rule rile 64 that specifies behavioral requirements of a specific network device.* In one embodiment, these behavioral requirements may be specified as a textual representation in the form of a decision tree.
> (Emphasis added.)

Contrary to the Examiner's assertion, Moir does *not* teach that the rule file is compiled and does *not* disclose or suggest that *one or more input rules comprise one or more **executable statements***.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements; and generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element, as required by independent claims 1, 20, and 21.

Claims 4 and 8

Claims 4 and 8 were rejected under 35 U.S.C. §102(e) as being anticipated over Moir. Regarding claim 4, the Examiner asserts that Moir discloses wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule a call chain emanating from the given rule (Moir: page 8, continued from rule file page 7; the component may have multiple operations with the same identifier but with different argument types; calling different operations). Regarding claim 8, the

Examiner asserts that Moir discloses wherein the step of determining which of the plurality of configuration elements could be accessed further comprise the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element (Moir: page 8, continued from rule file page 7; the component may have multiple operations with the same identifier but with different argument types; calling different operations).

Appellants note that the present application teaches, for example, that "the call chain can include calls by the rule to other items, sueh as additional rules and utility methods." (Page 3, lines 26-28.) Contrary to the Examiner's assertion, the fact that "the component may have *multiple operations* with the same identifier but with different argument types" does *not* teach a "call chain," as defined in the context of the present invention, and does *not* teach *a set of instance chain accesses for the given configuration element*. Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule *a call chain emanating from the given rule,* and does *not* disclose or suggest *a set of instance chain accesses for the given configuration element*.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule a call chain emanating from the given rule, as required by claim 4, and do not disclose or suggest wherein the step of determining which of the plurality of configuration elements could be accessed further comprise the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element, as required by claim 8.

Claim 9

Claim 9 was rejected under 35 U.S.C. §102(e) as being anticipated over Moir. Regarding claim 9, the Examiner asserts that Moir discloses wherein the given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class (Moir: page 6, paragraph 72; number of rules defined within a rule file), and wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements (Moir: page 8, continued from rule file page 7; invocation of more than operation with different arguments).

Contrary to the Examiner's assertion, "the invocation of more than operation with different arguments" does *not* teach *determining <u>every access</u> for the other configuration class to other configuration element.* Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest *determining <u>every access</u> for the other configuration class to other configuration element.*

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class, and wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements, as required by claim 9.

<u>Claims 14 and 16</u>

Claims 14 and 16 were rejected under 35 U.S.C. §102(e) as being anticipated over Moir. Regarding claim 14, the Examiner asserts that Moir discloses wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules (Moir: page 8, <operation> is similar to the definition in the spec that "deferred rules are invoked as a sequence, typically at the end of a configuration session"). Regarding claim 16, the Examiner asserts that Moir discloses wherein the at least one exeeutable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules (Moir: page 8, continued from rule file page 7; batch of operations with the same identifier but with different argument types).

Moir teaches the grammar of the body of a rule. Contrary to the Examiner's assertion, the <operations> disclosed by Moir does *not* teach that "deferred rules are invoked as a sequence, typically at the end of a configuration session" and does *not* teach *triggering output rules corresponding to the given input configuration element through <u>deferred triggering</u> of the one or more output rules.* Similarly, the operations with the same identifier (but with different argument types) disclosed by Muir does *not* teach *triggering output rules corresponding to the given input configuration element through* ***batch triggering*** *of the one or more output rules.* Moir teaches the grammar of the body of a rule, but does *not* disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules, and does *not* disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules.

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules, as required by claim 14, and do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules, as required by claim 16.

Claim 15

Claim 15 was rejected under 35 U.S.C. §102(e) as being anticipated over Moir. Regarding claim 15, the Examiner asserts that Moir discloses wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through direct triggering of the one or more output rules (Moir: page 9, paragraph 105).

In the text cited by the Examiner, Moir teaches:

> [0105] At block 86, the rule program 66 is loaded into the network connection device 12, responsive to a user (or manager) request. For example, the rule program 66 may be loaded into the network connection device 12 from a remote location on demand from a user or manager.

Thus, Moir teaches that the rule program 66 may be loaded into the network connection device 12 from a remote location on demand from a user or manager; Moir does *not* teach, however, *to* ***trigger output rules*** *corresponding to the given input configuration element through* ***direct triggering*** *of the one or more output rules.*

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through direct triggering of the one or more output rules, as required by claim 15.

Claim 17

Claim 17 was rejected under 35 U.S.C. 103(a) as being unpatentable over Moir and in view of Presley. Regarding claim 17, the Examiner acknowledges that Moir fails to teach performing a circularity check, but asserts that Presley teaches a method (that) further comprises the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself (page 4, paragraph 47) in order to provide reliable and predictable performance (page 1, paragraph 5).

In the text cited by the Examiner, Presley teaches:

> [0047] The core services 51 obtain the site-specific resource and component configuration XML doclets 53-55 by invoking probing service layers (not shown). The core services 51 analyze, advise, and correct out-of-bound and invalid configuration parameters and identify cyclic relationships. A set of new XML doclets 58 are generated and fielded to the individual network components 33 for implementation. The core services 51 also exports management, advising and alert services 56. The core services 51 can be exported through a browser service 57, as implemented on the management console 31 (shown in FIG. 2).

Thus, Presley teaches identifying cyclic relationships; Presley, however, does *not* teach performing a circularity check by *determining dependency relationships between two or more output rules* and by *determining whether a given one of the two or more output rules depends upon itself.*

Thus, Moir, Tuatini and Presley, alone or in combination, do not disclose or suggest the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself in order to provide reliable, as required by claim 17.

Conclusion

The rejections of the cited claims under sections 102 and 103 in view of Moir, Tuatini and Presley, alone or in any combination, are therefore believed to be improper and should be withdrawn. The remaining rejected dependent claims are believed allowable for at least the reasons identified above with respect to the independent claims.

The attention of the Examiner and the Appeal Board to this matter is appreciated.

Respectfully,

[illegible]

Date: December 14, 2009

Kevin M. Mason
Attorney for Applicant(s)
Reg. No. 36,597
Ryan, Mason & Lewis, LLP
1300 Post Road, Suite 205
Fairfield, CT 06824
(203) 255-6560

CLAIMS APPENDIX

1. In a system having a plurality of devices, wherein a plurality of configuration elements are associated with the plurality of devices, a method for automated generation of executable modules associated with the devices, the method comprising the steps of:

accessing information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules;

determining which of the plurality of configuration elements could be accessed based on the one or more input rules;

generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements; and

generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element.

2. The method of claim 1, wherein the one or more input configuration elements are described by one or more configuration classes and wherein the one or more input rules are described by one or more rule files.

3. The method of claim 1, wherein the step of determining which of the plurality of configuration elements could be accessed further comprises the step of determining read and write sets of configuration elements for a given one of the one or more rules.

4. The method of claim 3, wherein the step of determining read and write sets of configuration elements further comprises the step of determining for the given rule a call chain emanating from the given rule.

5. The method of claim 4, wherein the step of determining for a given rule a call chain emanating from the rule further comprises the steps of determining whether the given rule accesses one or more items and determining whether one or more other configuration elements are accessed by the one or more items.

6. The method of claim 5, wherein the one or more items comprise one or more rules or one or more utility methods.

7. The method of claim 5, wherein the step of determining read and write sets of configuration elements further comprises the steps of determining whether the one or more items accesses one or more additional items and determining whether one or more additional configuration elements are accessed by the one or more additional items.

8. The method of claim 1, wherein the step of determining which of the plurality of configuration elements could be accessed further comprise the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element.

9. The method of claim 8, wherein the given configuration element comprises a configuration element of a configuration class, wherein the given configuration element is another configuration class, and wherein the step of determining, for a given one of one or more configuration elements able to be accessed by an input rule, a set of instance chain accesses for the given configuration element further comprises the step of determining every access for the other configuration class to other configuration elements.

10. The method of claim 1, wherein the step of generating at least one executable module further comprises the step of generating at least one class for a given one of the one or more output rules, the at least one class defining the at least one executable module.

11. The method of claim 10, wherein the at least one class comprises one or more statements adapted to access at least one given configuration element that corresponds to the one

or more output rules.

12. The method of claim 10, wherein each of the at least one classes comprises one or more methods adapted to access the at least one given eonfiguration element.

13. The method of claim 12, wherein the access comprises reading, writing, or modifying the at least one given configuration element.

14. The method of claim 1, wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through deferred triggering of the one or more output rules.

15. The method of claim 1, wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through direct triggering of the one or more output rules.

16. The method of claim 1, wherein the at least one executable module is adapted to trigger the one or more output rules corresponding to the given input configuration element through batch triggering of the one or more output rules.

17. The method of claim 3, wherein the one or more output rules comprise two or more output rules, and wherein the method further comprises the step of performing a circularity check by determining dependency relationships between the two or more output rules and by determining whether a given one of the two or more output rules depends upon itself.

18. The method of claim 1, wherein the information further comprises at least one range restriction corresponding to the given input configuration element and wherein the at least one executable module is adapted to ensure that the at least one range restriction is met when the given configuration element accessed by the one or more triggered output rules is assigned a value.

19. The method of claim 1, wherein the information further comprises at least one referential integrity restriction corresponding to the given input configuration element and wherein the at least one executable module is further adapted to ensure that the at least one referential integrity restriction is met when the given configuration element is accessed by the one or more triggered output rules.

20. In a system having a plurality of devices, wherein a plurality of configuration elements are associated with the plurality of devices, an apparatus for automated generation of executable modules associated with the devices, the apparatus comprising:

a memory; and

at least one processor, coupled to the memory;

the apparatus being operative:

to access information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules;

to determine which of the plurality of configuration elements could be accessed based on the one or more input rules;

to generate one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements; and

to generate at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element.

21. An article of manufacture for use in a system having a plurality of devices, wherein a plurality of configuration elements are associated with the plurality of devices, and for automated generation of executable modules associated with the device, the article of manufacture comprising:

a machine readable medium containing one or more programs which when executed implement the steps of:

accessing information about one or more input configuration elements of the plurality of configuration elements, wherein the one or more input configuration elements are associated with one or more input rules;

determining which of the plurality of configuration elements could be accessed based on the one or more input rules;

generating one or more output rules using at least the accessed information, the accessed configuration elements, and the input rules, wherein an output rule corresponds to one or more input configuration elements and wherein said one or more input rules comprise one or more executable statements; and

generating at least one executable module adapted to access at least a given one of the input configuration elements and to trigger one or more of the output rules corresponding to the given input configuration element.

EVIDENCE APPENDIX

There is no evidence submitted pursuant to § 1.130, 1.131, or 1.132 or entered by the Examiner and relied upon by appellant.

RELATED PROCEEDINGS APPENDIX

There are no known decisions rendered by a court or the Board in any proceeding identified pursuant to paragraph (c)(1)(ii) of 37 CFR 41.37.